Maps in ES6:

The map() method in JavaScript is a built-in method that can be used to create a new array with the results of calling a provided function on every element in the calling array. In ES6 (ECMAScript 6), the map() method is available on the Array prototype and it can be used to transform any array-like object.

Here is an example of using the map() method to create a new array with the square of each element in an array:

```
let nums = [1, 2, 3, 4, 5];
let squaresOfNums = nums.map(function(number) {
  return number * number;
});
console.log(squaresOfNums); // Output: [1, 4, 9, 16, 25]
```

In this example, the map() method is called on the nums array, and it takes a callback function as an argument. The callback function is called for each element in the array, and it receives the current element as an argument. The function should return the new value for the current element.

You can also use an arrow function as callback function

```
let nums = [1, 2, 3, 4, 5];
let squaresOfNums = nums.map(number => number * number);
console.log(squaresOfNums); // Output: [1, 4, 9, 16, 25]
```

The map() method keeps the original array as it is. It creates a new array that contains the elements transformed by the callback function. This makes the method useful for creating a new array with the same length as the original array, but with different values for its elements.

It's important to note that the map method doesn't change the original array, it always returns a new array with the transformed elements.

Here is an example of creating a new Map and adding key-value pairs to it:

```
let person = new Map();
person.set("name", "John");
person.set("age", 35);
console.log(person.get("name")); // Output: "John"
console.log(person.get("age")); // Output: 35
```

In this example, a new Map object is created and the set() method is used to add key-value pairs to it. The set() method takes two arguments: the key and the value. The get() method is used to retrieve the value associated with a given key.

You can also create a Map and set the key-value pairs on creation:

```
let person = new Map([
    ["name", "John"],
    ["age", 30]
]);
console.log(person.get("name")); // Output: "John"
console.log(person.get("age")); // Output: 30
```

In this example, an array of key-value pairs is passed as the argument to the Map constructor, creating the map with the key-value pairs.

You can also use an object as the key, but keep in mind that javascript object keys are converted to strings, so it's not recommended to use objects as keys.

The Map object provides a convenient way to work with key-value pairs, and it has several useful methods, such as set(), get(), has(), delete(), and clear() for adding, retrieving, checking for the existence of, removing and clearing key-value pairs respectively.

Sets in ES6:

In JavaScript, a Set is a collection of unique values, which can be of any type. In ES6 (ECMAScript 6), the Set object is a built-in object that provides a convenient way to work with unique values.

Here is an example of creating a new Set and adding values to it:

```
let set = new Set();
set.add(10);
set.add(20);
set.add(30);
console.log(set.has(10)); // Output: true
console.log(set.has(40)); // Output: false
```

In this example, a new Set object is created and the add() method is used to add values to it. The add() method takes one argument, the value to be added. The has() method is used to check if the set has a specific value. It returns a boolean indicating whether the value is present or not in the set.

You can also create a Set and set the values on creation:

```
let set = new Set([1, 2, 3]);
console.log(set.has(1)); // Output: true
console.log(set.has(4)); // Output: false
```

In this example, an array of values is passed as the argument to the Set constructor, creating the set with the unique values.

The Set object provides a convenient way to work with unique values, and it has several useful methods such as add(), has(), delete(), clear() for adding, checking for the existence of, removing and clearing the set elements.